RPAガバナンスハンドブックに係る UiPath対応事例集

2019/10/28

UiPath株式会社
ビジネスコンサルティング室

はじめに

免責事項

1. 過剰な管理ルールの制定を回避する
2. 重点管理するプロセスを決める
3. 「野良ロボット」の発生を防ぐ
4. 品質で後悔しないワークフローを作成する
5. 本番環境を利用したテスト
6. プロセスの停止・障害に備える
7. ロボットが稼働する端末を管理する
8. RPA導入効果を上げるためにロボット専用IDを発行する
9. SOX対象業務にRPA適用を行う

はじめに

- 本書ではPwCあらた監査法人とUiPath株式会社共同で作成した「RPAガバナンスハンドブック」にて取り上げた課題に対してUiPath製品およびUiPath提供ドキュメントを活用する場合の解決策例を紹介する
- 具体的な解決策として、製品はOrchestartorを中心に実装方法を例示し、提供ドキュメントはUiPathの導入に関わる方法論やサンプル・テンプレートをまとめたものであるUiPath導入メソドロジーの使用方法を解説する

免責事項

- 本書の内容は、RPA導入組織の事例やUiPath株式会社の私見が含まれています。
- RPAの利用環境や利用形態は各組織により異なるため、本資料に記載してある対応事例の通りのガバナンスが適応可能であることを保証するものではありません。利用者の自己の責任において参考として利用してください。
- 本書に含まれる情報に基づき、RPAガバナンスの構築や見直し、評価を行ったことにより被った損失や損害について、UiPath株式会社は、いかなる責任や義務を負いません。
- 本書に記載されている全ての商標及びサービスマークは、ライセンスを有し、正当な権限に基づき使用する商標です。これらを無断で使用することは禁止します。

1. 過剰な管理ルールの制定を回避する ～利用形態に応じた管理ルールを制定する～

- ロボットやワークフローを適切に管理しながらRPAの短期開発や柔軟性といったメリットを享受するには、利用形態に応じたポイントを押さえたルール制定が必要となる
- 利用形態を観点ごとに分類し、管理の目的に沿ったルールを制定する

利用形態分類の観点（例）	利用形態のパターン	ルール策定のポイント
CoEの設置方式	■ 中央集権型 ■ 分散型	■ 中央集権型では全社共通のルールをCoEが制定する ■ 分散型ではルールはガイドとして作成・配布し各部門が自治を行う余地を残す
開発者	■ ユーザー（EUC） ■ プロフェッショナル ■ 上記両方	■ 開発者のスキルに応じ自動化の対象制限や開発標準を定める ■ 複数のレベルの開発者がいる場合レベルに応たルールを制定する
プロセス利用範囲	■ 個人利用 ■ 部門内 ■ 部門横断	■ 一つのプロセスの利用範囲が広くなるほど障害発生時のリスクが大きくなるため、品質管理や障害対応に関するルールをより詳細に制定する ■ プロセス利用状況を可視化するためのルールを制定する
RPA導入段階/規模	■ 導入初期（ロボット少数、Orchestratorなし） ■ 本格展開期（ロボット多数、Orchestratorあり）	■ RPAの規模が拡大するとともに保守運用の工数が増えるためOrchestratorや管理テンプレートを活用し管理を標準化する ■ 基盤やOrchestratorの運用管理ルールを制定する

1. 過剰な管理ルールの制定を回避する ～ルール制定例① 利用形態に応じた開発標準～

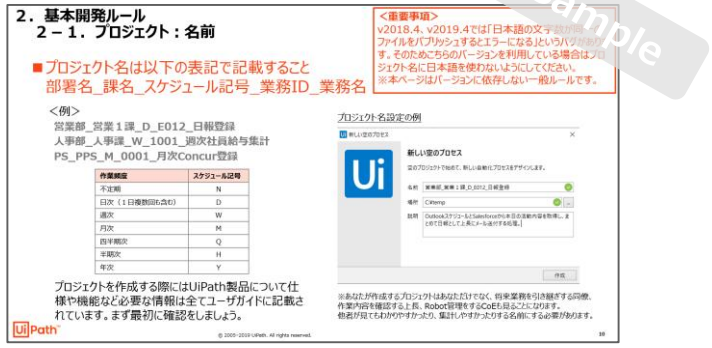
- 開発標準には利用形態に応じた内容を必要なレベルで記載する必要がある
- UiPath導入メソドロジーでは3つのレベルの開発標準を提供しており、利用形態に応じた使い分けが可能である

EUC Lv1開発ルール（開発初心者向け）

- 個人利用を前提としたワークフローに適用
- EUC向けの平易な開発標準
  1. プロジェクトの命名規則
  2. ワークフローの作成箇所、内部構成
  3. アクティビティの命名規則、配置
  4. 変数の型、命名規則
  5. ID/パスワードの管理
  6. 機密情報・個人情報の出力に関する禁止事項

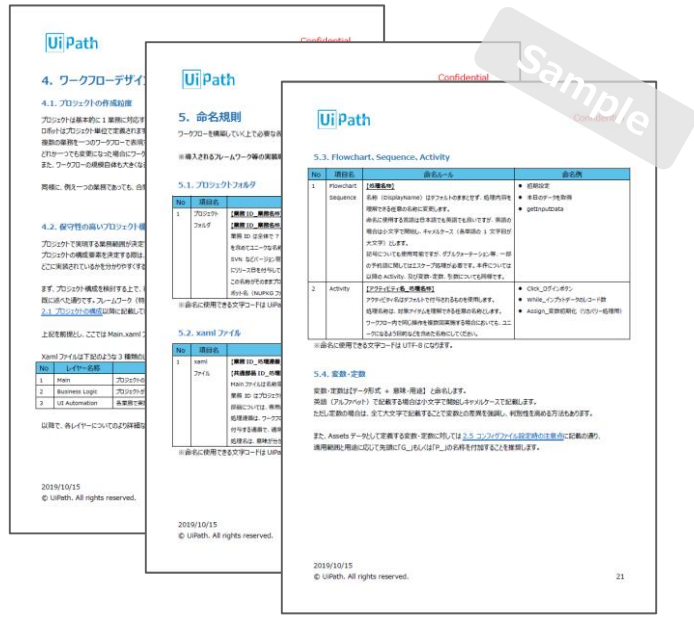
EUC Lv2開発ルール（開発中級者向け）

- 部門内で共有されるワークフローに適用
- Lv1の開発標準に以下を追加
  7. ログレベル・ログの書き方
  8. テストの確認事項と証跡
- Lv1、Lv2の開発標準はEUCを想定し分かりやすさを重視



プロフェッショナル開発標準（専門知識を有する技術者向け）

- 高リスクまたは部門を跨ぎ共有されるワークフローに適用
- 専門知識を持った開発者が作成を担当



1. 過剰な管理ルールの制定を回避する
～ルール制定例②　段階的な統制の強化～

- プロセスの共有がない状況や極めて小規模の段階において、ガバナンスに対して過度に構えることは不要である
- 規模や導入フェーズに応じ統制をとるべきタイミングで、必要なガバナンスを構築することを推奨する

イメージ

	プロセスの個人利用のみ	プロセスの部門内共有 （プロセス共有開始）	RPAの全社展開 （Orchestrator導入を想定） （Orchestrator導入）
導入段階のイメージ	• 1部署、RPA端末5台未満	• 1部署、RPA端末10台未満	• 複数部署、RPA端末10台以上
活用を推奨するドキュメント例	✓RPA禁止事項（次頁参照） ✓ライセンスリスト	✓UiPath導入マスタスケジュール ✓UiPath導入業務一覧 ✓開発手順定義書 ✓UiPath開発標準 ✓ライセンス管理方針	✓インフラロードマップ ✓環境基盤構築手順 ✓インフラ運用ガイド ✓アドミニストレーターガイド ✓運用設計ガイドライン ✓Orchestrator設定シート

1. 過剰な管理ルールの制定を回避する ～（参考）RPA導入初期の禁止事項の例～

- RPA導入規模が小さい場合であっても重大なインシデントを防ぐためのルールが必要となる
- 導入初期のRPA禁止事項の例を紹介する

フェーズ	禁止事項（例）	備考
業務選定	承認系操作	■ 承認ボタンを押すような操作は人が行うべき ■ 直前までの操作を自動化することは可
業務選定	外部メール送信	■ ドラフトの作成までを自動化し、人が確認後手動で送信
業務選定	Webへの書き込み	■ Webサービスとして提供される業務アプリの場合は、データの更新等の直前の入力までを自動化し、更新、送信等は人の確認後に手動実施 ■ 業務アプリ以外（SNS等）はすべて書き込み禁止
開発	開発・実行の無断委託	■ リソース管理やブラックボックス化を防ぐ目的から、他人に開発を依頼する場合は上長等の承認を得る ■ プロセスの実行結果には業務担当者が責任を持つ
開発	ID/PWの貸し借り	■ 権限管理上問題となるため自分のID/PWのみで開発・運用 ■ ワークフローへのID/PWのハードコーディングは禁止
運用	ライセンスポリシーに違反する使い方	■ Studio上の実行ボタンから本番業務を実行するなど

1. 過剰な管理ルールの制定を回避する ～（参考）RPA管理ルール項目例～

- Orchestratorを導入し、ユーザー開発・プロフェッショナル開発ともに全社的に行う場合のルールの項目例を紹介する
- 以下の項目を参考に、自社の都合に合わせたルールを制定するとよい

【RPA管理ルール項目（例）】

総則

- 本書の位置づけ
- RPA導入目標/目的
- RPA適用方針
- ロボットの位置づけ
- 管理体制（CoE）

案件企画

- 案件企画の全体像
- 起案・承認ルール
- 各ステップの詳細
  - 自動化対象業務/対象外業務
  - 開発部門
  - 開発優先順位（開発工数、効果見込み、その他）

開発

- 開発スケジュールと作成物
- 開発環境・本番環境の使い分け
- 開発に関わるIDの管理
- 開発標準・禁止事項
- テスト・リリース

運用

- 運用体制・役割分担
- ロボットの実行
- 稼働状況のモニタリング
- プロセスの修正・廃止の手続き
- 各種バージョンアップへの対応
- 障害発生時の対応方針
- 運用に関するIDの管理ルール
- Orchestratorの管理
- 関係者間の引継ぎ

2. 重点管理するプロセスを決める ～プロセス重要度の定義付けと継続的な見直し～

- プロセスが停止・誤処理した場合の業務への影響はプロセスにより異なるため影響度に応じた対策の整備が必要である
- プロセスの重要度の定義し、それに応じて開発品質や対応優先度を定める

プロセス重要度の定義

- プロセスの重要度はプロセスの停止・誤処理が発生した場合のリスクとプロセスの利用範囲で区分される
- 重要度が高いものは専門知識を有した技術者によって開発され、停止・誤処理が発生した際に優先的に対応される

高リスクの定義（例）

- ✓ 重要情報を取り扱う
- ✓ 社外へのメール送信を行う
- ✓ 承認・登録系の操作を行う
- ✓ Web書き込みを行う
- ✓ 手作業での代替不可
- ✓ システム負荷が大きい
- ✓ SOX対象業務である

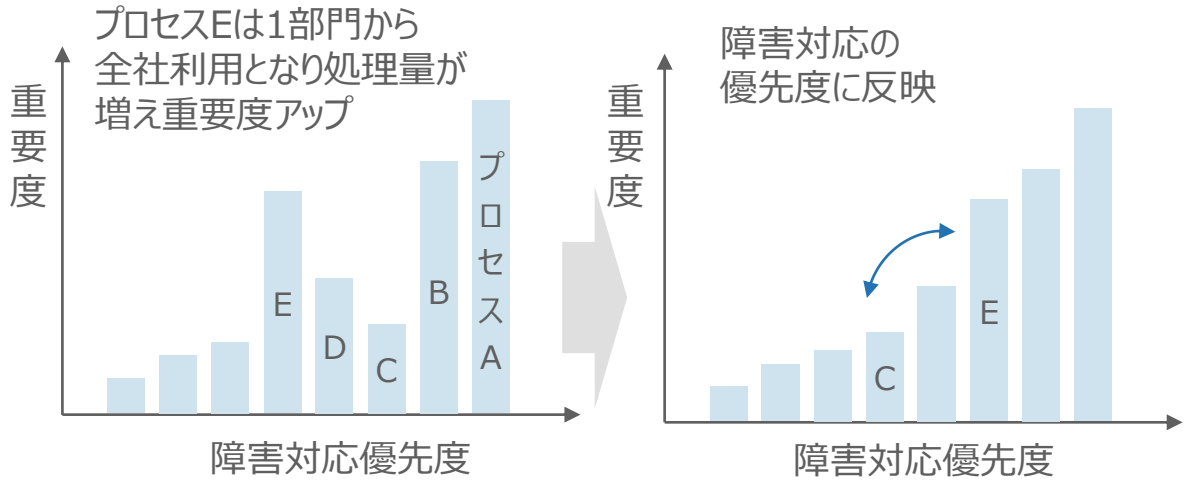
プロセス重要度の分類

	低リスク	高リスク
個人利用	低	高
部門内共有	中	高
部門横断	高	高

プロセス重要度の見直し

- 稼働監視ツールを用い利用部門や頻度・処理量を監視
- 必要に応じて重要度の見直しを行い、停止・誤処理の際の対応優先度に反映する

重要度・障害対応優先度の見直し



具体的な稼働監視の方法は次頁参照

2. 重点管理するプロセスを決める ～Orchestrator, Elasticsearch & Kibana による稼働状況監視～

- Orchestratorとログ分析・可視化ツールのElasticsearch & Kibanaを活用することで一覧性および視認性を向上することが可能となる

Orchestrator ダッシュボード

- ロボットの稼働状況・ジョブの実行結果をグラフィカルに表示
- トランザクションの統計情報をグラフィカルに表示
- ロボットの管理など各種メニューを表示

Elasticsearch & Kibana

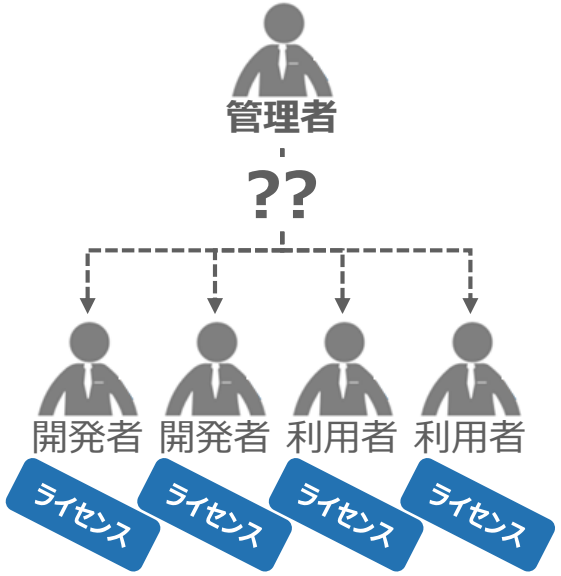
- Elasticsearchはオープンソースの分散型検索・分析エンジン
- KibanaはElasticsearchのデータを可視化するツール
- 端末稼働状況・エラー発生状況を詳細に把握可能
- カスタマイズ性が高い分、使いこなすには経験が必要

3. 「野良ロボット」の発生を防ぐ ～Orchestratorを活用したRPA資源管理～

- RPAの利用者が増加すると管理の目が行き届かない「野良ロボット」が作成・利用され、予期せぬインシデントが生じ得る
- Orchestratorを用いてライセンスとパッケージを一元管理することで「野良ロボット」の発生を防ぐことが可能となる

Orchestratorが無い場合の管理

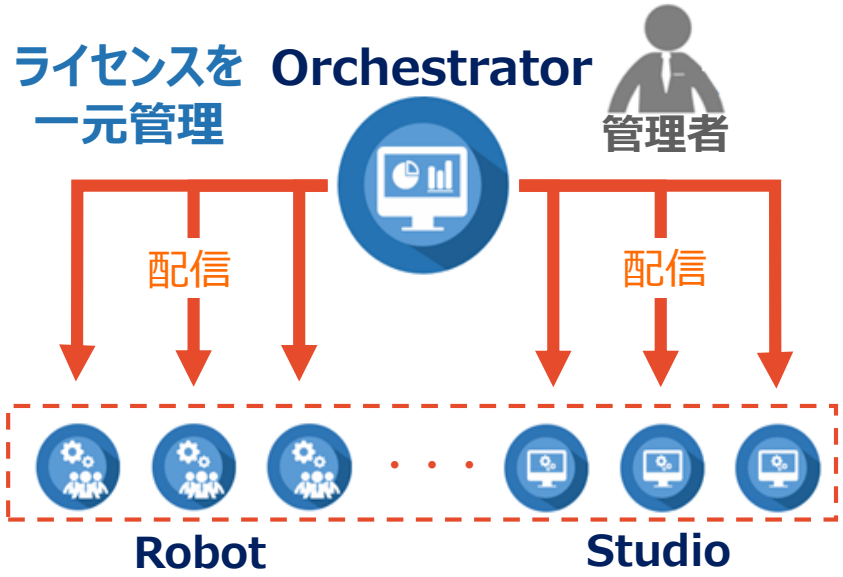
- ライセンスやパッケージの管理はユーザーに依存する
- 利用の実態の把握に多大な労力を要するため、スケールアップが困難になる



Orchestratorを活用したRPA資源の一元管理

ライセンスの管理

- OrchestratorからStudio/Robotのライセンスを配信することにより「野良ロボット」の作成・稼働を防ぐことが可能



パッケージ（開発成果物）の管理

- 開発者はパッケージをOrchestratorにアップロード
- 管理者がOrchestrator上でデプロイする端末を指定
- 利用者は管理者に指定されたパッケージのみ利用可能

Orchestratorの活用により少ない労力でスケールアップが可能

4. 品質で後悔しないワークフローを作成する ～UiPath導入メソドロジーサンプルの活用と継続保守～

- ワークフローの品質について適切な管理を行わない場合、エラーや保守工数が増加しプロジェクト全体に大きな手戻りを生じさせる恐れがある
- 品質を確保するためには導入メソドロジーのサンプルや可視化ツールを活用しながら開発と運用の二つの観点で対策を講じる

開発時の品質確保

- UiPath導入メソドロジーのサンプルやUiPath標準のフレームワークを活用する

UiPath開発標準	エラーが少なく保守性の高いワークフローを作成するためのルール
開発品質チェックリスト	ワークフローが開発標準通りに作成されているか確認するリスト
テスト計画書・仕様書	テストの実施方法や確認項目について定めたドキュメント
開発フレームワーク	開発工数をかけずに品質の高いワークフローを作成するための標準フレームワーク

継続的な品質向上

- ワークフローは一度リリースして品質管理を終了するのではなく継続的に稼働状況を監視し必要に応じて品質向上策を講じる

稼働状況モニタリング

Orchestrator

Elasticsearch & Kibana

品質向上策

分析結果をもとにワークフローを改修

エラーに関係するナレッジを集約し、以降のワークフロー開発にフィードバック

5. 本番環境を利用したテスト
～管理ルールの制定とテスト計画書・テスト仕様書の活用～

- 本番環境を使用してテストを実施すると、本番環境へテスト用データが混入される等業務に影響を及ぼすリスクがあるため、開発・検証用の環境を使用することを推奨する
- 開発・検証用の環境を用意できず本番環境を用いる必要がある場合は、業務に影響を及ぼさないための管理ルールを制定する
- UiPath導入メソドロジーの「テスト計画書」、「テスト仕様書」を活用し、事前の準備を行うことを推奨する

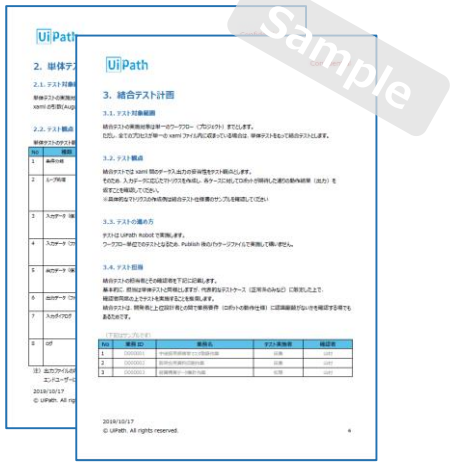
本番環境でのテスト実施に関するルールの例

関係者の合意	テストを実施する前に業務担当者とシステム担当者がテスト計画及びワークフローに業務影響を及ぼす可能性のある手順が含まれていないことを確認し合意する
立会い	テストには業務担当者が立ち会う
テスト用データ	テスト用データにはテスト用データであることを確認できるデータを用いる
更新の制限	テスト後に更新したデータを元に戻せない場合は更新処理を行わない範囲でテストを実施し、リリース後一定期間の試用期間を設け更新処理に問題がないことを業務担当者と開発者で確認する
データの復元	テスト後にデータの更新を元に戻せる場合はテスト終了後にデータを削除し、業務担当者がテスト前の状態に復元されたことを確認する
ログの保存	テスト時のログはテスト後に問題があった際に確認できるよう一定期間保存する

UiPath導入メソドロジーの活用

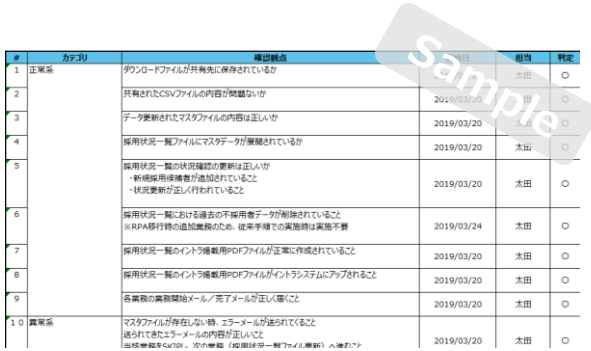
テスト計画書

- 一連の開発プロセスにおいて実施するテストの内容について定義したもので、対象範囲・進め方・観点・担当を記載する



テスト仕様書

- テストの具体的な手順を記載するドキュメントであり、以下が用意されている
  - 機能テスト仕様書
  - 単体テスト仕様書
  - UAT仕様書

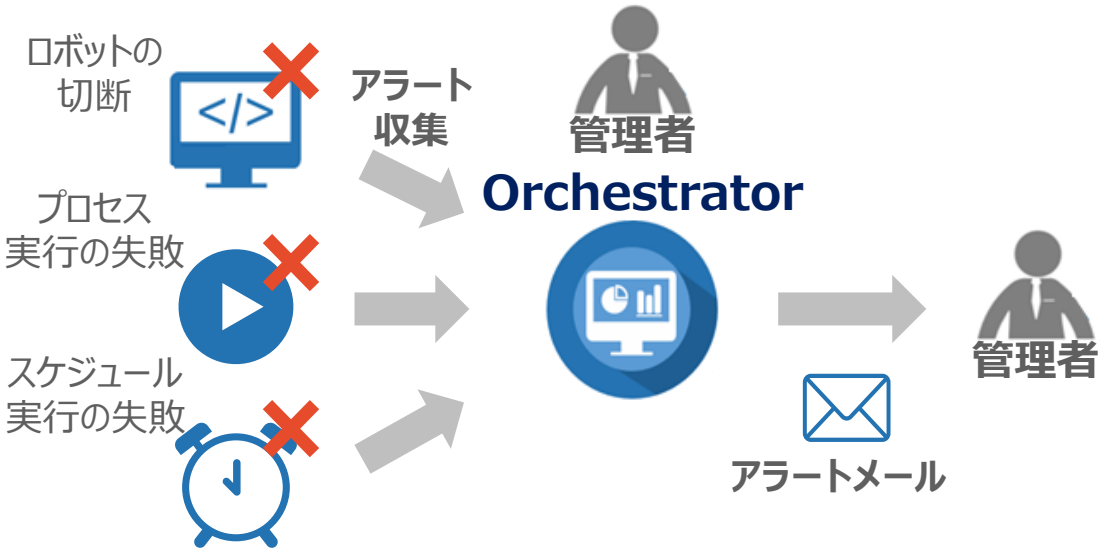


6. プロセスの停止・障害に備える ～アラート・運用方針定義書の活用～

- プロセスで障害が発生した際に素早く検知し業務とプロセスを復旧させるための体制や対応フローを事前に整備することで、業務への影響を軽減する
- OrchestratorのアラートUiPath導入メソドロジーの「運用方針定義書」を活用し、プロセスの障害に備える

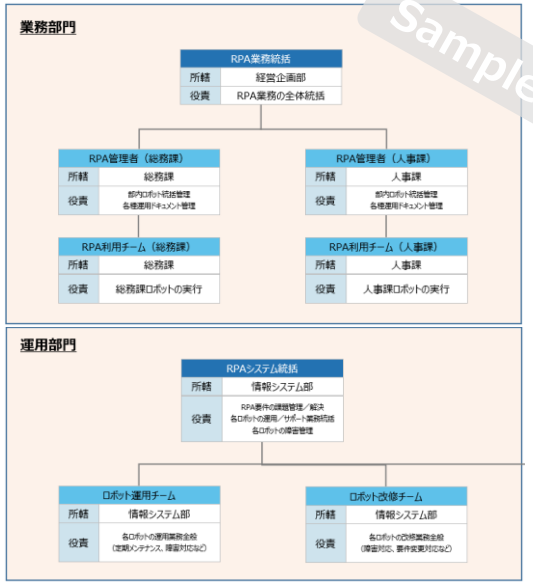
Orchestratorのアラート機能

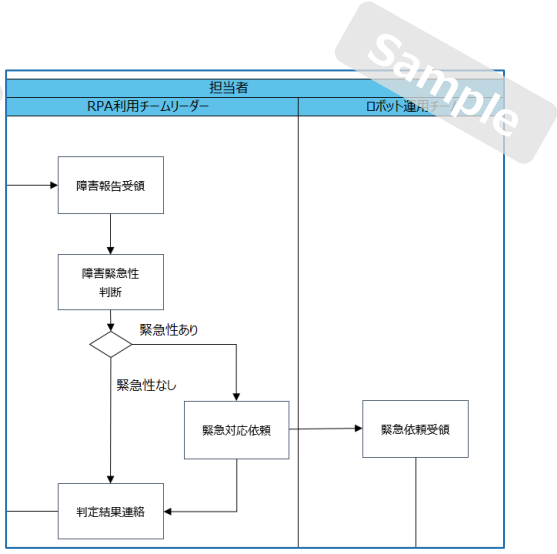
- アラート機能はプロセス実行の失敗やロボットの切断などの事象を重要度レベルと共に表示する機能であり、障害を検知する際に有用である
- 管理者はOrchestratorのアラートページから確認するか、メール配信を設定することで迅速に把握可能である（プロセスごとにメール配信先を設定する場合はワークフロー内に実装）



運用方針定義書

- UiPath導入メソドロジーの「運用方針定義書」の運用体制図を参考に体制と役割分担を明確にする
- 【障害対応】業務フローを参考に、障害検知後の連絡経路を事前に定めることで迅速な情報共有と対応が可能となる





7. ロボットが稼働する端末を管理する ～Unattended Robot の活用～

- 24時間365日稼働できるロボットのメリットを享受するために、無人ロボットが動作する端末のセキュリティを確保する工夫が必要となる
- Unattended RobotをOrchestratorと組み合わせて利用すると、負担を軽減したうえで安全な無人端末の運用が可能となる

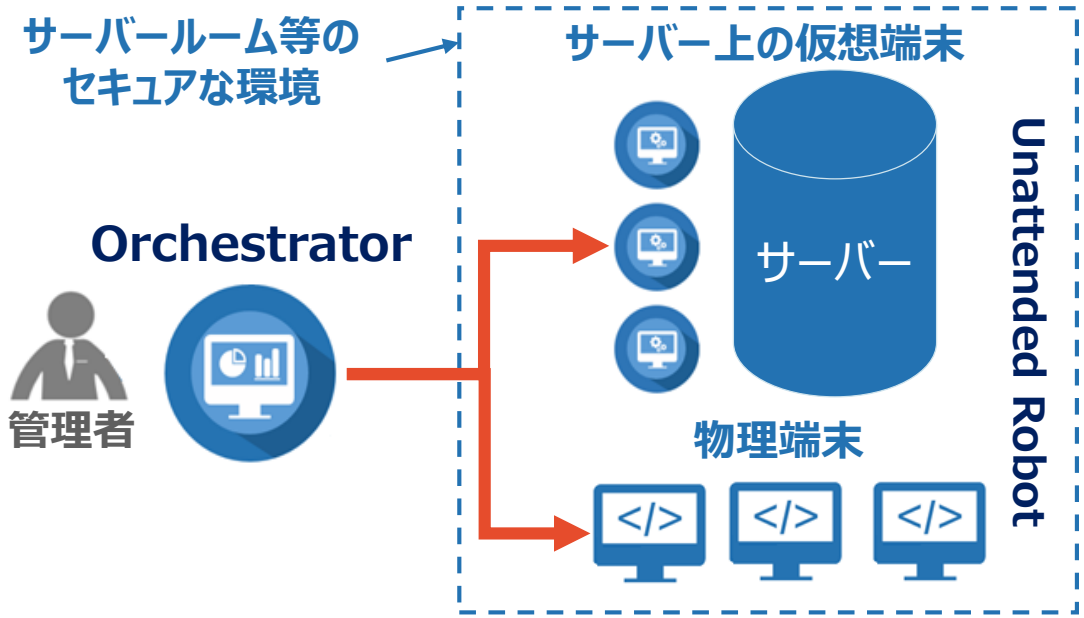
Unattended Robotの活用

- Unattended Robotは実行の際に人の関与が不要なロボットであり、効果的に利用することでROIの最大化に寄与するロボットである

稼働環境	■ ロボット専用の仮想端末・物理端末 ■ 画面の起動が不要なプロセスは画面をロックしたまま実行可能
実行方法	■ Orchestratorの管理画面から実行 ■ スケジュール起動、即時起動ともに可能
適する業務	■ 人間の関与が不要な業務 ■ 処理時間が長い業務 ■ スケジュール実行が可能な業務

稼働環境を隔離しての端末管理

- Unattended RobotはOrchestratorの管理画面より実行可能であるため、端末は物理的に安全な場所に保管しておく
- Orchestratorは端末の種類に依らず全ての端末を一元管理可能であり、管理工数の削減に貢献する

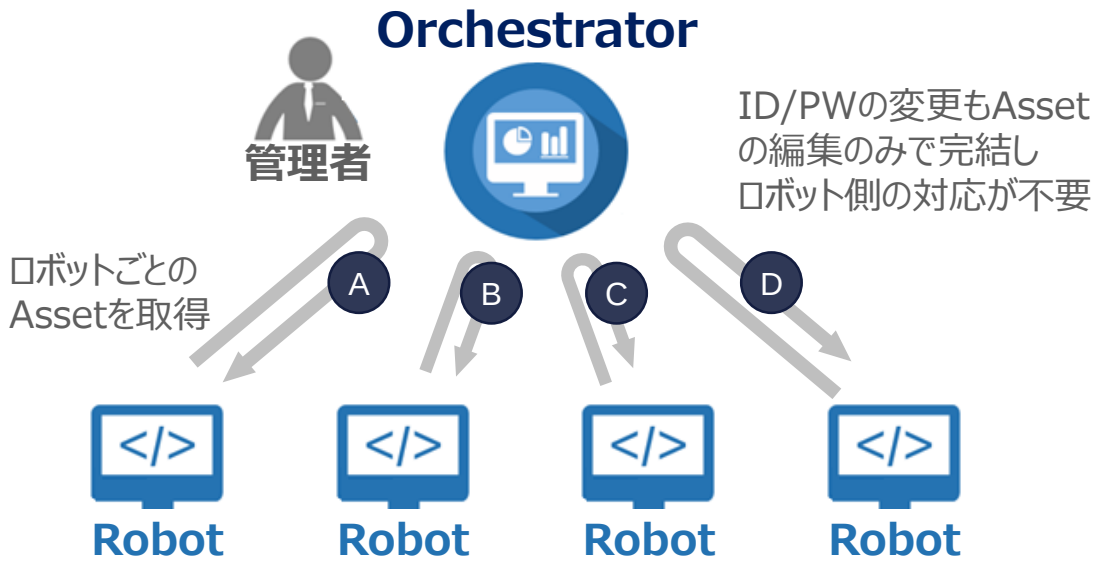


8. RPA導入効果を上げるためにロボット専用IDを発行する ～Orchestrator の Asset 管理機能の活用～

- 無人端末を活用しRPAの導入効果を得ようとする場合、ロボット用にシステム等の権限付与が必要になる。この際管理が不十分だとロボットの持つ権限の不正利用が生じる懸念がある
- OrchestratorのAsset管理機能により各種認証情報を一元管理することで、不正利用の抑止に寄与できる

Asset管理機能によるID/PWの集中管理

- Orchestrator上に登録するKey-Value型のユーザーデータであり、各種認証情報を集中管理可能
- ロボットごとにAssetを登録可能であり、登録した情報が他のユーザーに漏洩するリスクを低減できる

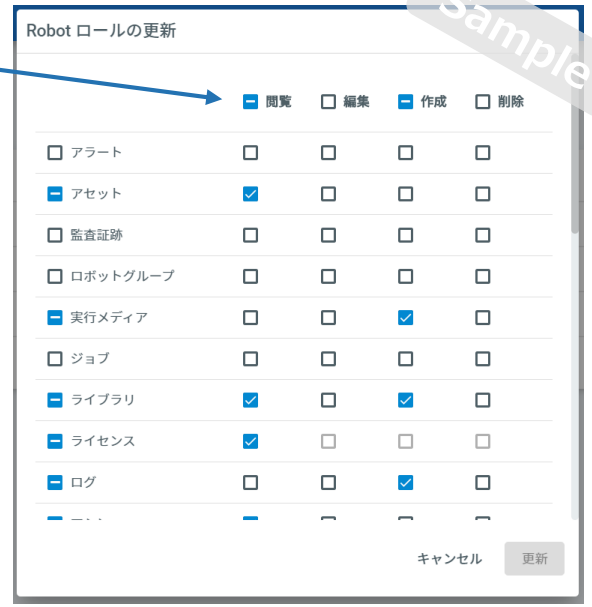


Orchestratorの管理権限

- Orchestrator上の各種管理項目はOrchestratorのユーザーごとに閲覧・編集等の詳細な権限設定が可能
- Assetに関しても無関係なOrchestratorユーザーによる情報の不正利用を予防できる

閲覧・編集・作成・
削除の4つの操作権限

機能ごとの
権限設定が可能



9. SOX対象業務にRPA適用を行う ～ポイントに応じた機能や管理テンプレートの活用～

- SOX対象業務にRPAを適用する場合は自動化後の業務プロセスがその業務に求められる要件を満たしていることを監査人と協議する必要がある
- Orchestratorや各種テンプレートを活用することで要件を満たすための工数を大幅に削減できる

SOX対象業務へのRPA適用のポイント

- SOX対象業務にRPAを適用する場合は主に以下のポイントに注意が必要である
- これまで例示した解決策を基に状況に合わせて実施することでSOX対応を試みる

適用時のポイント	UiPathを用いた解決策例
ポイント① SOXに影響を及ぼすプロセスか？	プロセス重要度を決める際にSOX対象業務であるかの確認項目を設けることでSOX対象業務プロセスの意図しない変更を防止する
ポイント② 当該プロセスは網羅的かつ正確に処理が行われているか？	要件定義書やテスト仕様書を活用することで要件漏れを防止し、Orchestratorのパッケージ管理を用いて管理外の変更を防止する
ポイント③ 当該プロセスが取り扱うデータは適切に保護されているか？	OrchestratorのAsset機能を用いて各種認証情報を管理し、システムやデータへの不正アクセスを防ぐ
ポイント④ RPAの管理体制・ルールはリスクを踏まえたものになっているか？全社へ浸透・定着しているか？	RPAルールの策定に加え、RPA資源を一元管理可能なOrchestratorを用いることで全社レベルの管理を実現する

Orchestratorの監査証跡

- Orchestratorのすべての操作は監査証跡として自動的に保存され、閲覧およびCSV出力が可能である
- 証跡にはユーザー、操作日時、操作対象、操作内容が記録される。これにより管理の実態を遡及可能となる
- 項目をソートして出力できるほか、各項目の詳細を確認可能